UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Gojtanić

Vohljač za SAS on-line komunikacijo

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Gojtanić

Vohljač za SAS on-line komunikacijo

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

MENTOR: IZR. PROF. DR. PATRICIO BULIĆ

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge razvijte vohljač, ki bo poslušal komunikacijo med igralnim avtomatom in centralnim računalnikom, ki uporabljata SAS on-line komunikacijo. Prebranim podatkom naj doda določene parametre in vse skupaj pošlje na tretji računalnik. Za implementacijo vohljača uporabite mikrokrmilnik ATMega 2560.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Andrej Gojtanić, vpisna številka 63990196, avtor zaključnega dela z naslovom:

Vohljač za SAS on-line komunikacijo (Sniffer for SAS on-line communication)

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Pivki, dne 27. junija 2016 Podpis študenta/-ke:

ZAHVALA

Največja zahvala gre mojim staršem, ki so mi študij omogočili. Podpirali so me vsa leta študija, čeprav nisem bil vzoren študent. Verjeli so vame tudi, ko sem večkrat padel na izpitu, pa tudi potem, ko sem moral ponavljati in pavzirati letnik, ker nisem opravil vseh obveznosti za vpis v naslednji letnik. Zato še enkrat hvala. Nato se bom zahvalil tudi mentorju za vso podporo pri izvedbi projekta. Nazadnje, ampak nič manj pomembno, se zahvaljujem vsem sošolcem za pomoč pri učenju in razumevanju snovi, ker mi samemu verjetno ne bi uspelo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
TCn	Timer/Counter n; n = 1, 2, 3…	Časovnik n; n = 1, 2, 3…
SAS	Slot Accounting System	Samonadzorni sistem
ALE	Arithmetic Logic Unit	Aritmetično logična enota
AVR	Automatic Voltage Regulator	
RISC	Reduced Instruction Set Computer	
PC1	Gaming machine computer	računalnik igralnega avtomata
PC2	SAS on-line computer	SAS on-line računalnik
PC3	Computer showing the results	Računalnik, ki prikazuje podatke
USART	Universal Serial Asinchronous Receiver/Transmitter	Univerzalni serijski asinhroni sprejemnik/oddajnik
USB	Universal Serial Bus	Univerzalno serijsko vodilo
RS232	Standard for serial communication, transmission of data	Standard za serijski prenos podatkov
R/W	Read/Write	Bralno/pisalni

Povzetek

Naslov: Vohljač za SAS on-line komunikacijo

V okviru diplomskega dela je bilo potrebno razviti vohljač, ki posluša komunikacijo med dvema računalnikoma. Prebranim podatkom doda določene parametre in vse skupaj pošlje na tretji računalnik. Za to smo razvili vezje, ki je priključeno neposredno na komunikacijski kabel, preko katerega potujejo podatki med dvema računalnikoma. Na vezju je mikrokrmilnik, ki prebrane podatke shrani, jim doda vse parametre ter vse skupaj pošlje na tretji računalnik. Na njem se izvaja program, ki je prav tako bil razvit za ta projekt, v katerem so podatki prikazani kronološko. Na ta način lahko vidimo, kje in kdaj je prišlo do napake.

Ključne besede: vohljač, SAS on-line sistem, RS232 komunikacija, USB, razhroščevanje.

Abstract

Title: Sniffer for SAS on-line communication

In the thesis, a sniffer had to be developed, which is listening the communication between two computers. It saves all the data, includes some other parameters and sends everything to a third computer. For this, a circuit board was developed, that is directly connected to the communication line used by the two computers. There is a microcontroller on it, which stores all the read data, appends all of the parameters and sends everything to the third computer. An application that was also developed for this project is running on the third computer, which displays the data received chronologically. In such a way we can find when and why an error happened.

Keywords: sniffer, SAS on-line, RS232 communication, USB, debugging.

Poglavje 1 Uvod

V podjetju kjer delam, se razvijajo in proizvajajo avtomatski igralni avtomati, kot so na primer rulete in video igralni avtomati. Ker večina igralnic za boljši nadzor nad dogajanjem priključi igralne avtomate na on-line sistem, je bilo v podjetju potrebno razviti tudi aplikacijo, ki se zna pravilno sporazumevati z njim. V projektu smo se osredotočili na SAS (Slot Accounting System) on-line sistem, ker smo ugotovili, da pri komunikaciji prihaja do določenih problemov in jih samo z beleženjem nismo bili sposobni rešiti. Zato smo se odločili razviti vezje, ki se priključi neposredno na komunikacijsko linijo in bere podatke, ki se po njej prenašajo, ter jih shranjuje. Pri tem k vsakemu shranjenemu byte-u doda še čas, ki je pretekel od sprejetja prejšnjega, ne glede na izvor, in vse skupaj pošlje na tretji računalnik. Na ta način bomo lahko videli, zakaj in kako pride do problema.

Za lažjo izdelavo smo se odločili, da bomo vezje razvili na osnovi Arduino MEGA2560 [4] razvojne ploščice, ker je prijazna za uporabo. Na njej je mikrokrmilnik Atmel ATMega2560, ki ima štiri vmesnike USART (komponente za komunikacijo) ter dovolj časovnikov (za štetje časa) za realizacijo. V projektu so uporabljeni trije vmesniki USART in pet časovnikov. Dva vmesnika USART sta uporabljena za branje podatkov, vsak za svoj izvor, tretji pa za pošiljanje podatkov. Poleg same razvojne ploščice se je naredilo še dodatno vezje, na katerem je čip MAX232, za branje podatkov s komunikacije RS232, ki jo uporablja on-line sistem, dve LED diodi, ki prikazujeta branje podatkov, ter mostiček, s katerim se določa hitrost povezave s tretjim računalnikom (115200bps ali 1Mbps).

Poglavje 2 Opis komunikacije SAS on-line sistema

V igralništvu se uporablja SAS On-line sistem za komunikacijo med igralnim avtomatom in centralnim računalnikom. Preko tega se lahko spremlja dogajanje na igralnem avtomatu, odklepa in zaklepa igralna mesta, pregleduje zgodovino dogodkov, nakazuje in odvzema kredite na igralnih mestih itd. Kar je za projekt zanimivo, so lastnosti povezave. Pogovor med računalniki poteka tako, da centralni računalnik vsake t milisekund (v večini primerov 40ms) pošlje poizvedbo igralnemu aparatu, ta pa odgovori. Vsi dogodki, ki se zgodijo na igralnem avtomatu, se shranijo v FIFO medpomnilnik in se eden po eden pošiljajo na centralni računalnik, ko ta pošlje poizvedbo. Torej za vsako poizvedbo po en dogodek. Če igralni avtomat 5 sekund ne dobi poizvedbe, začne sam pošiljati poizvedbo centralnemu računalniku, dokler se spet ne sinhronizirata.

2.1 Lastnosti komunikacije

- »Wakeup« način,
- hitrost 19200 bps,
- 1 start bit,
- 8-bitni podatki,
- 9. »wakeup« bit,
- 1 stop bit.

2.1.1 »Wakeup« način

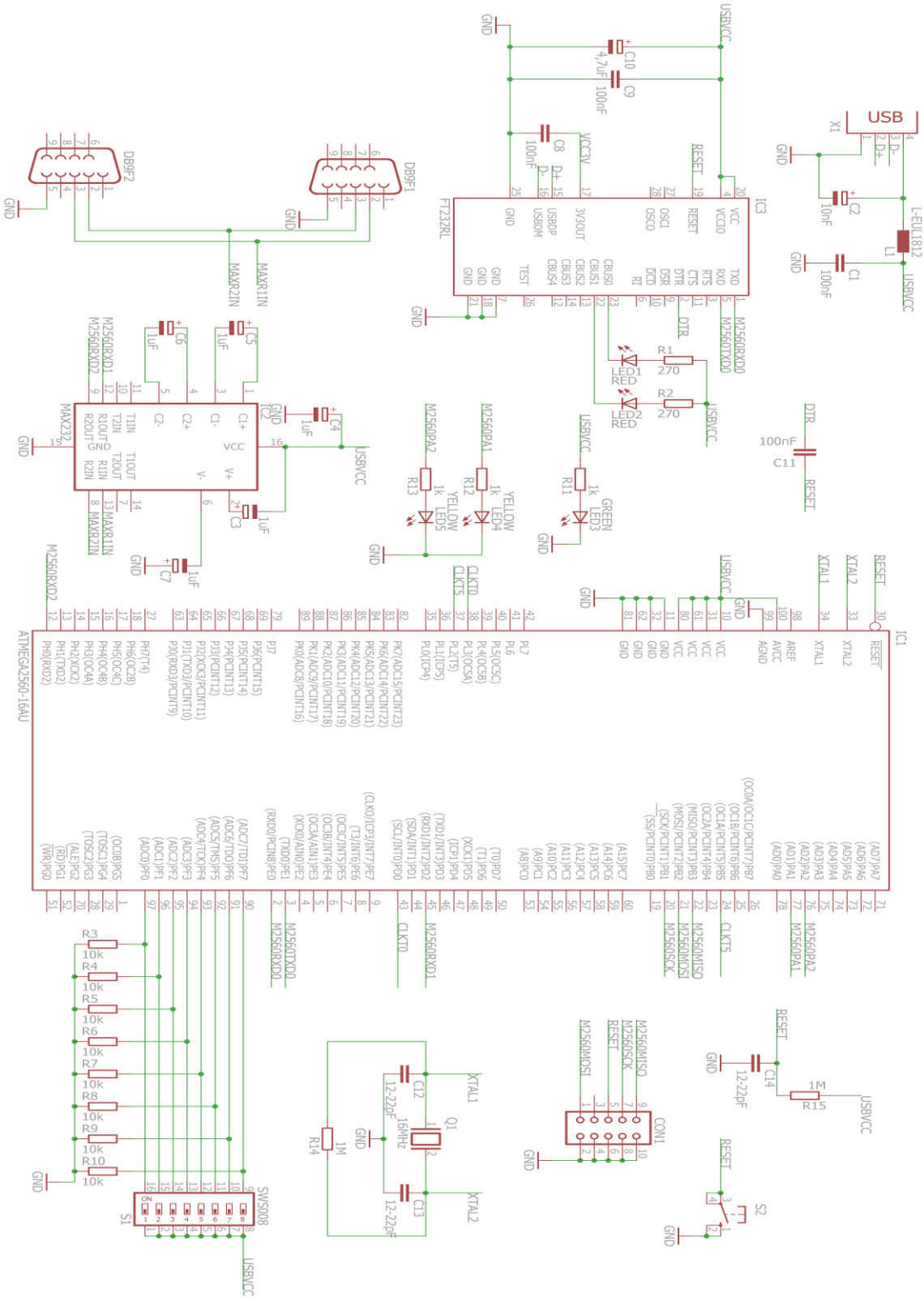
Pri tem načinu se 9. bit postavi na 1 pri poslanem prvem byte-u, da prejemnik ve, kje se paket podatkov začne.

Pri SAS on-line sistemu se deveti bit postavi na 1 samo pri poizvedbah. Torej ko strežnik pošlje poizvedbo, je pri prvem byte-u deveti bit enak 1, in ko igralni avtomat pošlje poizvedbo, je pri prvemu byte-u deveti bit enak 1. Ko igralni avtomat odgovarja, se deveti bit ne postavi.

Poglavje 3 Opis izdelanega vezja

Izdelano vezje je precej preprosto. Glavna komponenta je mikrokrmilnik Atmel ATMega2560 [1], na kateremu se izvaja program. Ker RS232 povezava uporablja drugačne nivoje kot USART vmesniki mikrokrmilnika, je potreben pretvornik. V projektu je za to nalogo uporabljen čip MAX232CPE [3] proizvajalca MAXIM. Prav tako je za USB povezavo potreben čip FT232RL [2] proizvajalca FTDI. Preko USB povezave vezje dobi napajanje (5V). Poleg tega ima vezje še dva DB9 konektorja, reset gumb, pet LED diod ter 10-pinski flat konektor za programiranje mikrokrmilnika. RESET linija je vezana na DTR vmesnik od FT232RL preko kondenzatorja. To nam omogoča programsko resetiranje vezja. Osem stikal dip-switch je preko pull-down uporov vezanih na PORTF. V projektu je uporabljen le eden, ki določa hitrost povezave (0=115200bps, 1=1Mbps). Ostali so dodani za bodoči razvoj vezja.

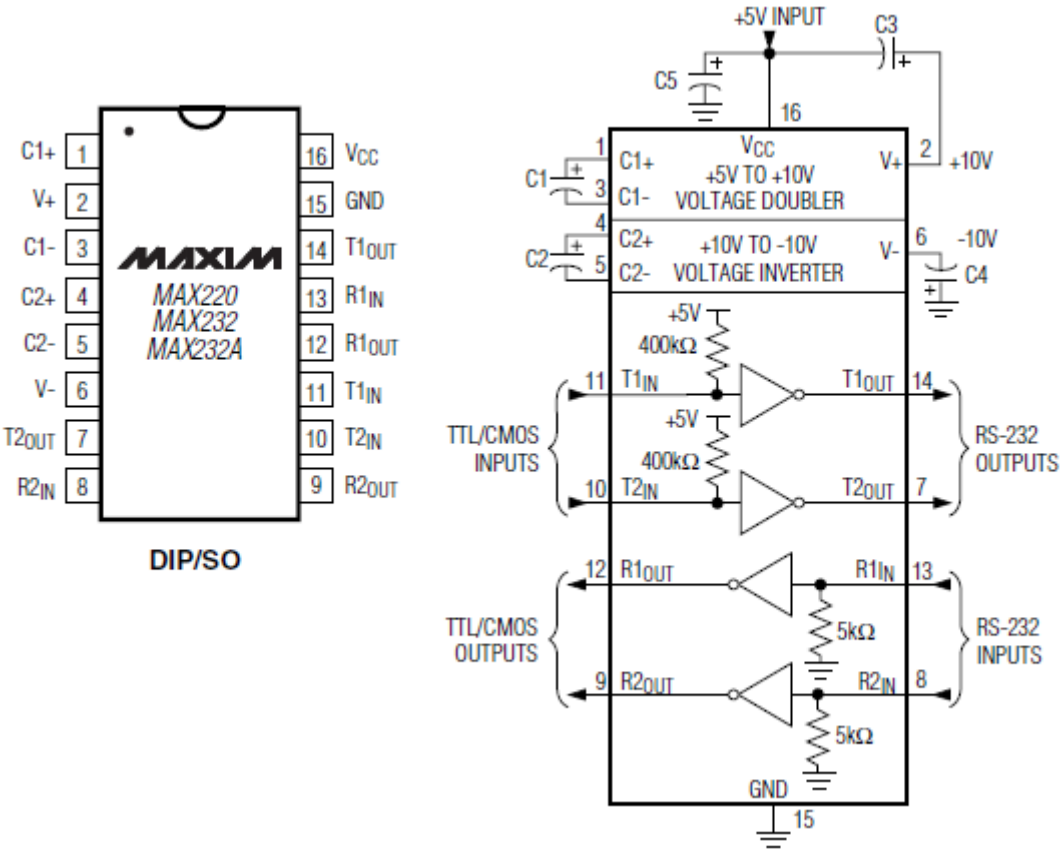
3.1 Shema vezja



Slika 3.1: Shema vezja

3.2 MAX232CPE

Čip MAX232CPE [3] se uporablja za pretvorbo signalov med CMOS/TTL in RS232. Ima možnost priklopa dveh linij (Rx in Tx), tako da za projekt zadostuje en čip. Za pravilno delovanje potrebuje 5 V napajalne napetosti, ki jo dobi od USB povezave, ter pet 1 uF elektrolitskih kondenzatorjev (C1 – C5). Na spodnji sliki je prikazano, kakšna mora biti vezava.



Slika 3.2: Vezava MAX232CPE [3]

3.3 FT232R

Čip FT232R [2] se uporablja kot vmesnik pri komunikaciji med vmesnikom USART in USB. Čip je tudi prenastavljen za krmiljenje LED diod, ki utripajo, medtem ko poteka komunikacija. Diodi sta povezani na vmesnika CBUS0, 1, ki sta že prenastavljena za krmiljenje diod. Poleg tega je za normalno delovanje potrebnih nekaj kondenzatorjev, kot je prikazano v shemi vezja na strani 6.

3.4 Mikrokrmilnik ATmega2560

Ker je opis mikrokrmilnika bolj obsežen, je opisan v svojem poglavju (glej Poglavje 4).

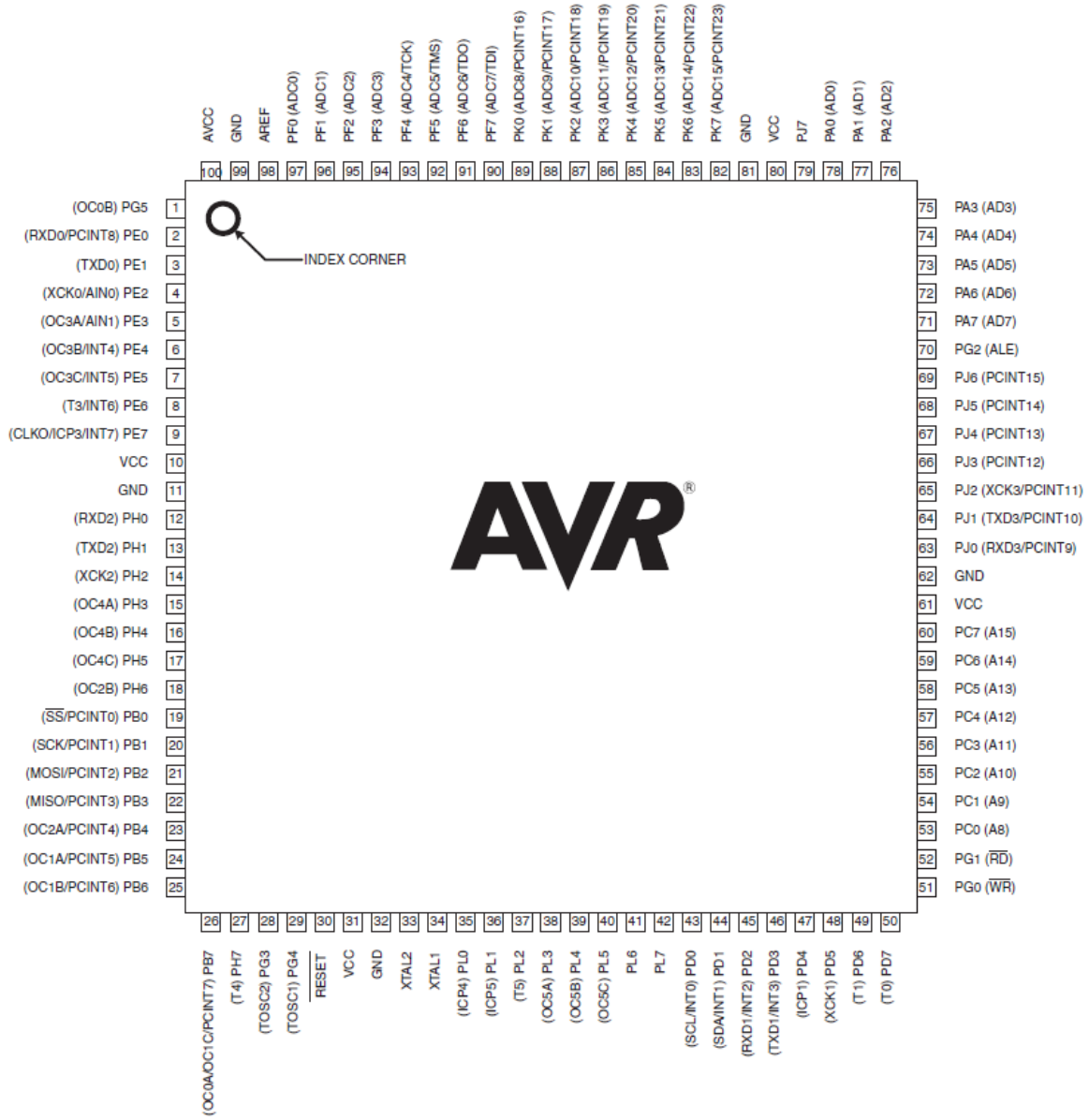
Poglavje 4 Mikrokrmilnik ATmega2560

V opisu mikrokrmilnika [1] so naštete in opisane enote in lastnosti, ki so pomembne za projekt in/ali bile v njem uporabljene.

4.1 Lastnosti

- Visoko performančni, 8-bitni AVR mikrokrmilnik z nizko porabo energije.
- Napredna RISC arhitektura.
- Visoko-vzdržljivi segmenti trajnega pomnilnika:
  - 256 KB sistemskega, samoprogramabilnega bliskovnega pomnilnika,
  - 4 KB EEPROM,
  - 8 KB notranjega SRAM-a.
- Periferne enote
  - dva 8-bitna časovnika z lastnima delilnikoma ure in možnostjo delovanja s primerjanjem,
  - štirje 16-bitni časovniki z lastnimi delilniki ure in možnostjo delovanja s primerjanjem ter zajemanjem,
  - štirje 8-bitni PWM kanali,
  - štirje programabilni serijski USART vmesniki.
- Možnost programskih in zunanjih prekinitev.

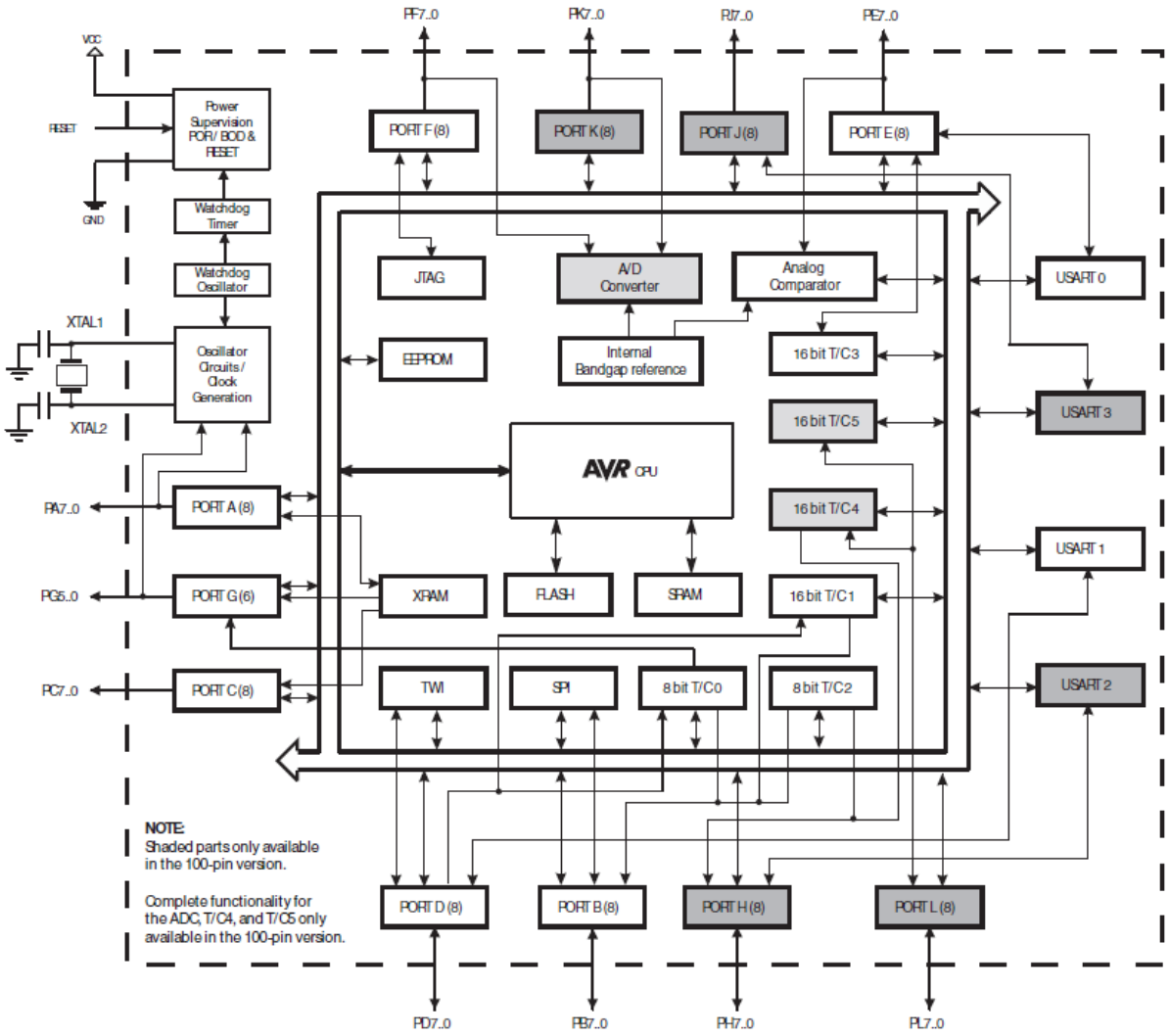
4.2 Konfiguracija vmesnikov



Slika 4.1: ATmega2560 TQFP-pinout [1]

ATmega2560 je 8-bitni CMOS mikrokrmilnik z nizko porabo energije, baziran na AVR RISC arhitekturi. Ker se večina ukazov izvede v eni urini periodi, se hitrost delovanja približa 1 MIPS-u na MHz. Vseh 32 8-bitnih delovnih registrov je povezanih z ALE enoto, tako da se lahko dostopa do dveh različnih registrov v eni urini periodi.

4.2.1 Blok diagram



Slika 4.2: Blok diagram ATmega2560 [1]

4.3 Opis vmesnikov

Opisani so vmesniki, ki so v projektu uporabljeni:

- VCC.

Napajalna napetost (+5 V).

- GND.

Zemlja (0 V)

- Port A (PA7..PA0).

Port A je 8-bitni Vhodno/Izhodni vmesnik z notranjim pull-up uporom. V projektu sta uporabljena vmesnika PA1 in PA2 kot izhoda za krmiljenje LED diod. Lahko se uporabi tudi za posebne funkcije.

- Port B (PB7..PB0).

Port B je 8-bitni Vhodno/Izhodni vmesnik z notranjim pull-up uporom. V projektu so uporabljeni vmesniki PB1 (SCK), PB2 (MOSI), PB3 (MISO) za programiranje mikrokontrolerja ter PB5 za generiranje urinega signala za časovnik TC5. Urin signal generira časovnik TC1.

- Port D (PD7..PD0).

Port D je 8-bitni Vhodno/Izhodni vmesnik z notranjim pull-up uporom. V projektu sta uporabljena vmesnika PD2 in PD7. PD2 je uporabljen kot vhod (RXD1) za vmesnik USART1 za prisluškovanje računalniku PC1, PD7 pa kot vhod, preko katerega dobi časovnik TC0 urin signal.

- Port E (PE7..PE0).

Port E je 8-bitni Vhodno/Izhodni vmesnik z notranjim pull-up uporom. V projektu je uporabljen vmesnik PE1 kot izhod (TXD0) za vmesnik USART0, ki se uporablja za USB komunikacijo z računalnikom PC3.

- Port F (PF7..PF0).

Port F je uporabljen kot vhod. Nanj so priključena stikala dip-switch. Z njihovo različno konfiguracijo lahko določamo razne lastnosti delovanja mikrokontrolerja. V projektu je trenutno uporabljen samo eden. Z njim določamo hitrost USB povezave.

- Port H (PH7..PH0).

Port H je 8-bitni Vhodno/Izhodni vmesnik z notranjim pull-up uporom. V projektu je uporabljen vmesnik PH0 kot vhod (RXD2) za vmesnik USART2 za prisluškovanje računalniku PC2.

- RESET.

Reset vhod. Če ta vmesnik postavimo na nizek nivo za dlje, kot je minimalna dolžina pulza (2,5 us), sprožimo reset na mikrokrmilniku.

- XTAL1.

Vhod do invertirajočega ojačevalnika oscilatorja in vhod do enote notranje ure.

- XTAL2.

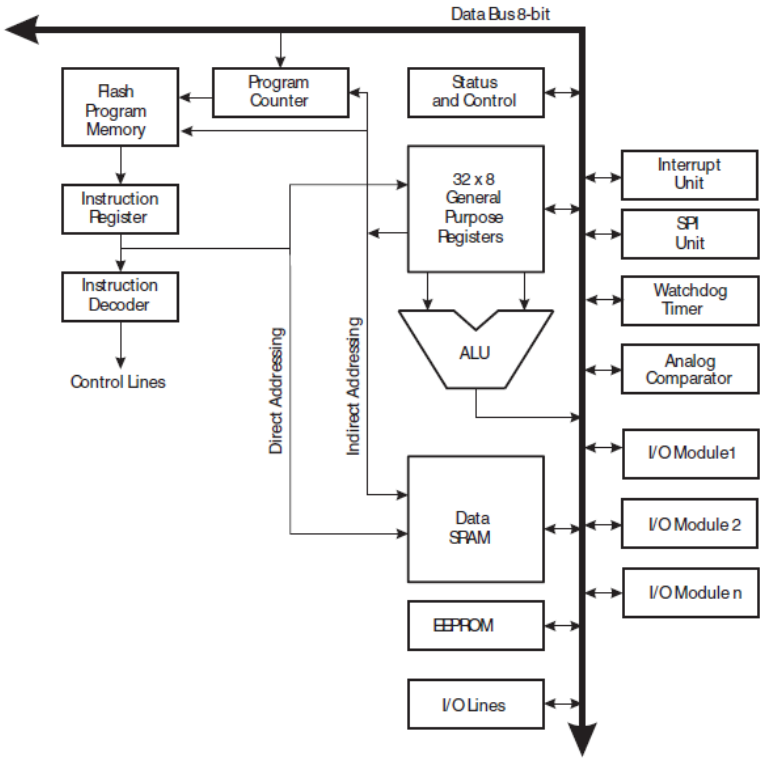
Izhod invertirajočega ojačevalnika oscilatorja.

- AVCC.

Vmesnik za napajanje Porta F in A/D pretvornika. Priporočeno ga je eksterno vezati na VCC, tudi če ne uporabljamo A/D pretvornika.

4.4 Jedro CPE

4.4.1 Blok diagram

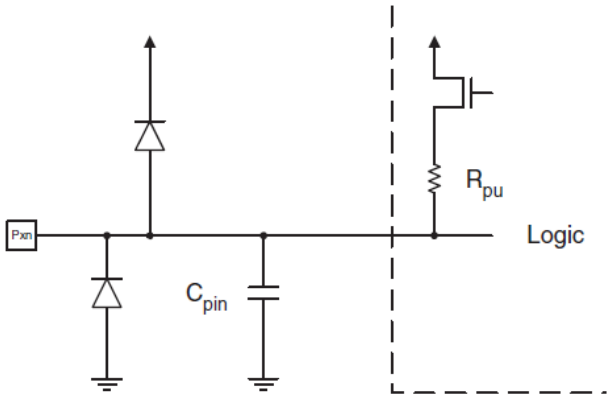


Slika 4.3: Blok diagram CPE jedra [1]

Da bi maksimizirali performanse in paralelizem, AVR uporablja Harvardsko arhitekturo – ločen pomnilnik za ukaze in operande. Vsebuje eno-nivojski cevovod, kar pomeni, da se naslednji ukaz bere iz pomnilnika, medtem ko se trenutni ukaz izvršuje. Na tak način se vsako urino periodo izvrši en ukaz.

4.5 V/I vmesnik

Vsi V/I vmesniki se lahko programirajo posamično. Za vsak vmesnik posebej lahko določimo, ali bo vhod ali izhod. Če je definiran kot vhod, lahko vklopimo ali izklopimo pull-up upor tudi za vsak port posebej. Porti so dovolj močni za krmiljenje LED diod. Vsak port ima tudi zaščitno diodo na VCC in GND, kot je prikazano na sliki 1.4.



Slika 4.4: V/I vmesnik [1]

Vsi registri in reference na bite so napisani splošno. Mali x pomeni številno črko vmesnika, mali n pa pomeni številko bita. Na primer PORTB3 se nanaša na bit 3 v vmesniku B, kar bo tukaj napisano kot PORTxn. Za vsak vmesnik imamo 3 registre:

- podatkovni register PORTx (R/W): če je vmesnik konfiguriran kot vhod, s spreminjanjem vrednosti PORTxn vklapljamo (1) in izklapljamo (0) pull-up upor. Če je vmesnik konfiguriran kot izhod, s spreminjanjem vrednosti PORTxn določamo, ali bo na njem visok nivo (1) ali nizek nivo (0).
- Smerni register (Data Direction Register) DDRx (R/W): če postavimo DDRxn na 1, bo vmesnik Pxn izbran kot izhod, drugače pa kot vhod.
- Port Input Pins PINx (R): z branjem tega registra izvemo vrednost vmesnikov (0 ali 1).

Pri tem projektu ima večina V/I vmesnikov posebne funkcije, razen PA1 in PA2, ki služita za krmiljenje LED diod. Funkcije ostalih vmesnikov so opisane v poglavju 4.3.

4.6 Časovniki (Timer/Counter)

Časovnik (TC) je enota za štetje časa. Ima več načinov za štetje in tudi več možnosti izbire urinega signala. Poleg tega se lahko urin signal deli, tako da šteje počasneje. Časovniki TC0, 1, 5 so uporabljeni za štetje časa med sprejetimi byte-i, časovniki TC3, 4 pa za krmiljenje LED diod na vmesnikih PA1 in PA2. Poleg štetja, časovnika 1 in 5 generirata urin signal. Časovnik 1 za časovnik 5, časovnik 5 pa za časovnik 0. Na ta način dobimo 24-bitni (uporabljenih 22) časovnik, ki šteje, ne da bi pri tem uporabljal procesorski čas.

4.6.1 Opis registrov

Opisani so registri, ki so za projekt pomembni.

4.6.1.1 TCCRnA (TC Control Register A)

Bit	7	6	5	4	3	2	1	0
(0x80)	COMnA1	COMnA0	COMnB1	COMnB0	COMnC1	COMnC0	WGMn1	WGMn0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Slika 4.5 TCCRnA kontrolni register [1]

- Bit 7 : 6 – COMnA1:0: Compare Output Mode for Channel A.
- Bit 5 : 4 – COMnB1:0: Compare Output Mode for Channel B.
- Bit 3 : 2 – COMnC1:0: Compare Output Mode for Channel C (časovnik TC0 nima).

Ti biti določajo obnašanje OCnx vmesnikov. Te funkcije prevladajo nad normalnimi funkcijami vmesnikov. V vsakem primeru pa je treba v DDRx registru tem vmesnikom določiti pravilno smer. Obnašanje teh vmesnikov je odvisno tudi od WGMnx bitov. V projektu imamo dve različni nastavitvi:

- COMnA1:0 = [0,0] (časovniki TC0, 3, 4).

V tem načinu so vmesniki OCnA neaktivni. Port ima normalno funkcijo.

- COMnA1:0 = [1,1] (časovnika TC1, 5).

V tem načinu časovnik postavi vmesnik OCnA na visok nivo, ko TCNTn doseže vrednost OCRnA, in nazaj na nizek nivo, ko gre TCNTn nazaj na 0. Na ta način generira urin signal, medtem ko šteje.

- Bit 1 : 0 – WGMn1:0: Waveform Generation Mode.

Skupaj z bitoma WGMn3:2 v TCCRnB registru določajo štetje časovnika, izvor MAX časovnika (do katere vrednosti šteje) ter kakšno bo generiranje signala na izhodih.

V spodnji tabeli je razvidno, kakšni načini delovanja so na voljo.

Mode	WGMn3	WGMn2	WGMn1	WGMn0	Timer/Counter Mode of Operation	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Tabela 4.1: Načini delovanja časovnikov [1]

Načini delovanja, uporabljeni v projektu, so naslednji:

- 4. CTC (časovniki TC0, 3, 4).

V tem načinu časovnik šteje od 0 do vrednosti, ki je zapisana v OCRnA registru, nato začne znova. Ko doseže vrednost OCRnA, lahko sproži prekinitev, kar je tudi uporabljeno pri vseh treh časovnikih.

- 5. Fast PWM, 8-bit (časovnika TC1, 5).

V tem načinu časovnik šteje od 0 do 0xFF. Pri tem primerja TCNTn z OCRnA. Ko TCNTn doseže vrednost OCRnA, spremeni nivo na vmesniku OCnA glede na to, kako so nastavljeni biti COMnx. Na ta način generira urin signal.

4.6.1.2 TCCRnB – (TC Control Register B)

Bit	7	6	5	4	3	2	1	0
(0x81)	ICNCn	ICESn	–	WGMn3	WGMn2	CSn2	CSn1	CSn0
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Slika 4.6 TCCRnB kontrolni register

- Bit 4 : 3 – WGMn3:2: Waveform Generation Mode.

Ta dva bita skupaj z bitoma WGMn1:0 v registru TCCRnA določata način delovanja, opisan v poglavju 4.8.1.1.

- Bit 2 : 0 – CSn2:0: Clock Select.

Ti biti določajo, kateri urin signal bo TC uporabljal za štetje.

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Tabela 4.2: Izbira urinega signala za časovnik (Clock Select) [1]

V projektu imamo tri različne nastavitve:

- CSn2:0 = [0,1,0] (časovnik TC1).

V tem primeru časovnik uporablja sistemsko uro deljeno z 8 (clk/8 = 2 MHz). Na tak način se časovnik poveča za 1 vsake 0,5 us, kar je pri danih hitrostih komunikacije dovolj natančno. Časovnik TC1 je namreč najnižji časovnik, zato določa hitrost štetja in neposredno natančnost vseh treh vezanih časovnikov (TC1-TC5-TC0).

- CSn2:0 = [1, 0, 0] (časovnika TC3, 4).

Enako kot v prejšnjem primeru, le da je tukaj sistemski urin signal deljen z 256 (clk/256 = 62,5 kHz).

- CSn2:0 = [1, 1, 0] (časovnika 0, 5).

Izbrana nastavitev določa, da je urin signal zunanji in da se števec poveča na padajoči del.

4.6.1.3 TCNTnH and TCNTnL – (TimerCounter)

Bit	7	6	5	4	3	2	1	0
(0x85)	TCNTnH[15:8]							
(0x84)	TCNTnL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Slika 4.7 TCNTn register (števec).

Pri časovnikih TC1, 3, 4, 5 je to 16-bitni register/števec, pri časovniku TC0 pa 8-bitni. V obeh primerih se števec prebere v eni urini periodi, čeprav je vodilo 8-bitno, ker ima pri 16-bitnih časovnikih 8-bitni TEMP register, kamor se shrani TCNTnH byte, medtem ko se TCNTnL prebere.

4.6.1.4 OCR1nH and OCR1nL – (Output Compare Register A)

Bit	7	6	5	4	3	2	1	0
(0x89)	OCRnAH[15:8]							
(0x88)	OCRnAL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Slika 4.8 OCRnA register.

Ta register je pri časovniku TC0 8-bitni, pri ostalih pa 16. TCNTn se v vsaki urini periodi primerja z OCRnA. Ob dogodku TCNTn = OCRnA se lahko sproži prekinitev (časovniki TC0, 3, 4) ali se uporablja za generiranje valovanja (časovnika TC1, 5 za clk).

4.6.1.5 TIMSKn – (TC Interrupt Mask Register)

Bit	7	6	5	4	3	2	1	0
(0x6F)	–	–	ICIEn	–	OCIEnC	OCIEnB	OCIEnA	TOIEn
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

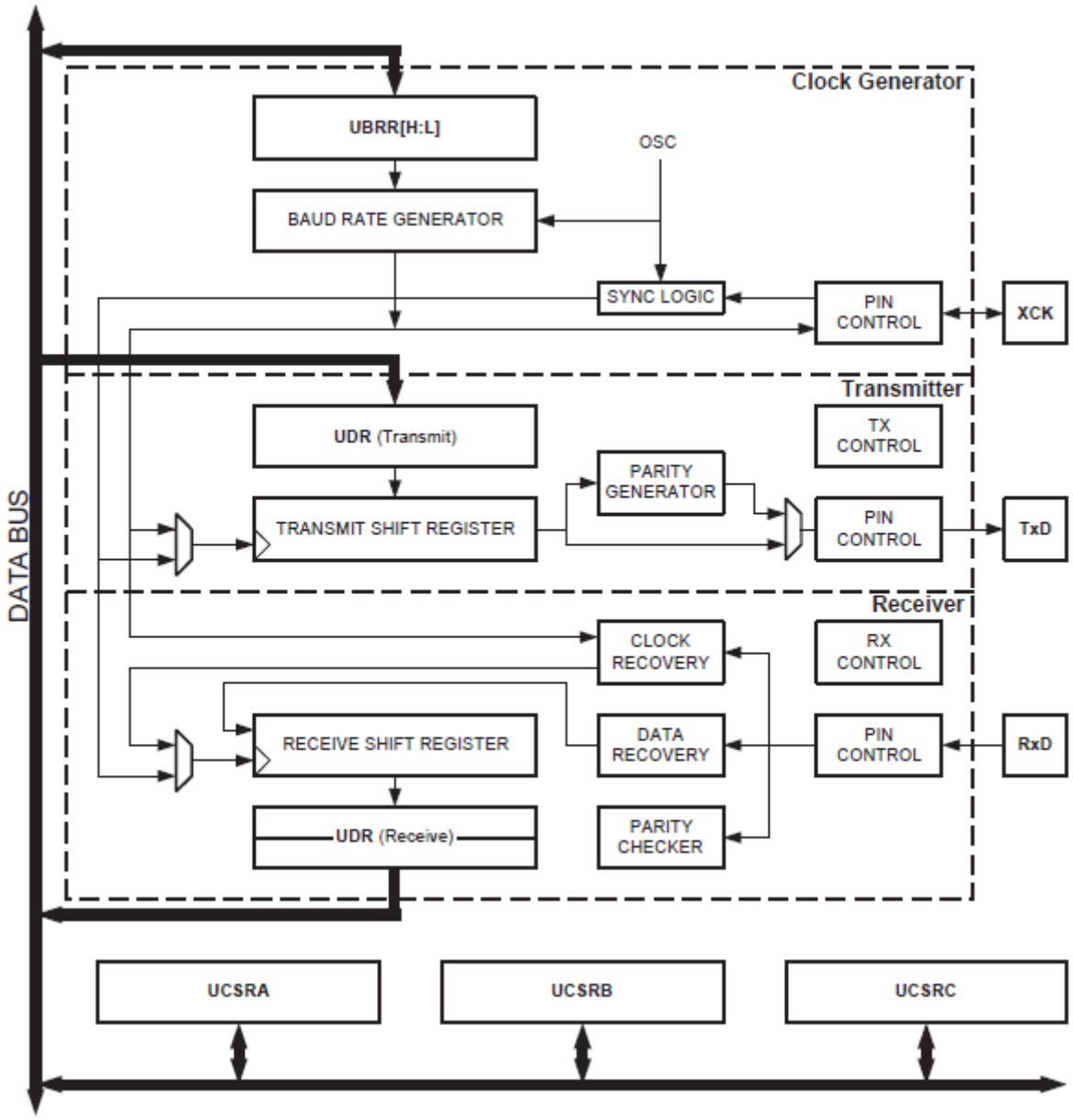
Slika 4.9 TIMSKn register

- Bit 3 – OCIEnA: Output Compare A Match Interrupt Enable (časovniki TC0, 3, 4).

Če je ta bit postavljen na 1, so omogočene prekinitve ob dogodku TCNTn = OCRnA (Compare Match).

4.7 Vmesnik USART

Vmesnik USART se uporablja za komunikacijo. V projektu so uporabljeni trije vmesniki, USART1 in 2 (RS232) za prisluškovanje ter USART0 (USB) za pošiljanje podatkov. Na spodnji sliki je prikazan blok diagram USART vmesnika.



Slika 4.10 Blok diagram USART vmesnika [1]

Črtkani pravokotniki ločujejo tri glavne enote (z vrha): generator urinega signala (Clock Generator), oddajnik (Transmitter) in sprejemnik (Receiver). Kontrolne registre si delijo vse enote. Oddajnik lahko pošilja pakete podatkov brez presledkov. Podpira štiri vrste komunikacije: normalna asinhrona, double-speed asinhrono, master-sinhrono in slave-sinhrono. V projektu je uporabljena normalna asinhrona. Baud-rate generator se uporablja za generiranje urinega signala. Z vpisom določene vrednosti v register UBRRn, določimo hitrost komunikacije. Glej poglavje 4.8.4.5.

4.7.1 Formati podatkovnih paketov (frame)

USART vmesniki podpirajo vse kombinacije naslednjih lastnosti:

- 1 start bit
- 5, 6, 7, 8, ali 9-bitni podatki
- brez, soda, liha pariteta
- 1 ali 2 stop bita

V tem projektu imamo pri RS232 delu 1 start bit, 9-bitne podatke, brez paritete in 1 stop bit. Pri pošiljanju 9-bitnih podatkov je potrebno deveti bit postaviti, preden se pošlje podatke v UDRn register. Enako velja pri branju. Deveti bit je treba prebrati preden se prebere UDRn. Pri USB povezavi pa imamo 8-bitni podatkovni del.

4.7.2 Opis Registrov

4.7.2.1 UDRn – (USART I/O Data Register)

Bit	7	6	5	4	3	2	1	0
	RXB[7:0]							
	TXB[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Slika 4.11 UDRn register [1]

Skupni register za sprejemanje in oddajanje. Sprejema se preko RXB in oddaja preko TXB dela. V UDRn se lahko piše samo, kadar je UDREn zastavica v UCSRnA registru postavljena na 1. Karkoli se piše, ko zastavica ni postavljena, bo ignorirano s strani vmesnika USART. Podatek se iz UDRn prenese v oddajni register (Transmit Shift Register), ko je ta prazen, in nato se serijsko pošlje na TxDn pin.

4.7.2.2 UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn
Read/Write	R	R/W	R	R	R	R	R/W	R/W
Initial Value	0	0	1	0	0	0	0	0

Slika 4.12 UCSRnA kontrolni register

- Bit 7 – RXCn: USART Receive Complete.

Če je zastavica postavljena na 1, pomeni, da je v UDRn neprebran podatek. Lahko se uporabi za generiranje prekinitev.

- Bit 6 – TXCn: USART Transmit Complete.

Zastavica se postavi, ko se iz oddajnega registra (Transmit Shift Registra) pošlje celoten okvir in ni nobenih podatkov v UDRn registru. Zastavico lahko uporabimo za proženje prekinitev.

- Bit 5 – UDREn: USART Data Register Empty.

Zastavica pove, če je UDRn prazen in pripravljen na sprejem novih podatkov. Lahko generira prekinitev.

4.7.2.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

Slika 4.13 UCSRnB kontrolni register [1].

- Bit 7 – RXCIEn: RX Complete Interrupt Enable n.

Če postavimo bit na 1, omogočimo prekinitev na RXCn zastavici. Pri tem morajo biti globalno omogočene prekinitve v registru SREG (Status Register).

- Bit 4 – RXENn: Receiver Enable n.

S postavitvijo bita na 1 omogočimo sprejemanje podatkov.

- Bit 3 – TXENn: Transmitter Enable n.

S postavitvijo bita na 1 omogočimo pošiljanje podatkov.

- Bit 2 – UCSZn2: Character Size n.

Skupaj z bitoma UCSZn1:0 v registru UCSRnC določajo velikost podatkovnega dela okvirja, ki se uporablja pri komunikaciji.

- Bit 1 – RXB8n: Receive Data Bit 8 n.

Ta bit predstavlja 9. bit podatkovnega dela paketa pri sprejemanju. Prebrati ga je treba, preden se prebere UDRn.

4.7.2.4 UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	0

Slika 4.14 UCSRnC kontrolni register [1]

- Bits 7 : 6 – UMSELn1:0 USART Mode Select.

S temi biti določamo vrsto komunikacije, in sicer po spodnji tabeli.

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)

Tabela 4.3 Vrste komunikacije [1]

V projektu je povsod izbrana asinhrona komunikacija.

- Bits 5 : 4 – UPMn1:0: Parity Mode.

S temi biti izbiramo uporabljeno pariteto. V projektu je izbrana komunikacija brez paritete, zato sta oba bita postavljena na 0.

- Bit 3 – USBSn: Stop Bit Select.

Če bit postavimo na 1, se bosta poslala 2 stop bita na koncu paketa. V projektu je postavljen na 0, ker imamo komunikacijo z enim stop bitom.

- Bit 2 : 1 – UCSZn1:0: Character Size.

Skupaj z bitom UCZn2 v UCSRnB registru določajo velikost podatkovnega dela okvirja glede na spodnjo tabelo.

UCSZn2	UCSZn1	UCSZn0	Character SIze
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Tabela 4.4: Velikost podatkov [1]

V projektu imamo pri dveh vmesnikih USART (USART1, 2) 9-bitni podatkovni del, zato so vsi trije biti postavljeni na 1. Pri USART0 pa imamo 8-bitni podatkovni del, tako da sta bita UCSZn0:1 na 1.

4.7.2.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8
	–	–	–	–	UBRR[11:8]			
	UBRR[7:0]							
	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

Slika 4.15 UBRRn register [1]

Z vpisom vrednosti v ta register določamo hitrost (Baud Rate) komunikacije. V projektu imamo tri različne hitrosti. Pri 19200 bps je vrednost UBRRn 51, pri 115200 je 8, pri 1 Mbps pa 0, kar je razvidno iz spodnje tabele.

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4K	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2K	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8K	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4K	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6K	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8K	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2K	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4K	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250K	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

Tabela 4.5: Vrednoti UBRRn pri 16MHz [1]

Poglavje 5 Opis programa

Program je napisan tako, da za večino operacij uporablja prekinitve. Uporabljen je tak način, ker je v projektu najbolj pomemben čas med dvema zaporednima sprejetima byte-oma. S prekinitvami dosežemo to, da se lahko ob vsakem trenutku program prekine ter izvede prekinitveno rutino. V njej si zapomni čas, ki je pretekel od prejšnjega sprejetega byte-a. Mislim, da je pri taki rešitvi najmanj odstopanja od realnih časov med byte-i. V glavni (main) funkciji se najprej izvede inicializacija vmesnika USART, ki je povezan na priključek USB s hitrostjo 115200 bps, da se pošlje zaglavja. Za tem se inicializirajo časovniki, določi se hitrost USB povezave, inicializira vmesnik USART za prisluškovanje ter vstopi v neskončno while(1) zanko. V njej se primerjata indeksa Head, ki označuje prvi prazen prostor v krožnem medpomnilniku, in Tail, ki označuje prvi še ne poslani byte v krožnem medpomnilniku. Če sta različna, kar pomeni, da je v krožnem medpomnilniku eden ali več novih okvirjev za poslati na računalnik PC3, se kliče funkcija Send(), v kateri se pošilja byte za byte-om, dokler Tail ne doseže Head. Neskončna zanka while(1) in funkcija Send() se lahko prekineta kadarkoli. Tako je zagotovljena največja natančnost pri shranjevanju vrednosti časovnikov.

Format okvirja, ki se pošlje na računalnik PC3 (4 x 8 bitov za vsak sprejeti byte):

1.

bit	7	6	5	4	3	2	1	0
definicija	bit 9 pri sprejetem podatku	izvor (0 = PC1, 1 = PC2)	vrednost TC0					

2.

bit	7	6	5	4	3	2	1	0
definicija	vrednost TC5							

3.

bit	7	6	5	4	3	2	1	0
definicija	vrednost TC1							

4.

bit	7	6	5	4	3	2	1	0
definicija	sprejeti podatek							

5.1 Opis kode programa

V tem delu je opisan vsak blok programa posebej. Ker so nekateri bloki enaki, razlikujejo se samo po indeksih, bo opisan eden, pri drugemu pa bo samo napisano, kar je različno.

5.1.1 Zaglavja (Header)

Zaglavja so strukture, ki vsebujejo določene podatke o samem vezju ter programu, ki se na njem izvaja. Preko njih se vezje ter program, ki prikazuje podatke, poslane z vezja, sporazumeta. To je potrebno zato, ker se bo lahko vezje spreminjalo ali celo naredilo novo/drugačno, in bo preko zaglavij program vedel, s kom se pogovarja. Program za prikazovanje je razvit tako, da je lahko nadgradljiv za prihodnje verzije programa na samem vezju ali pa z drugimi vezji. Spodaj je en primer zaglavja za boljše razumevanje.

Definicija zaglavja, ki pove lastnosti povezave:

```
struct Header_Connection
{
        uint_least32_t Start;
        uint_least8_t HeaderTyp;
        uint_least8_t HeaerVer;
        uint_least8_t HeaderLen;
        uint_least8_t NextHeadLen;

        uint_least32_t BaudRate;
        uint_least8_t DataBits;
        uint_least8_t Parity;
        uint_least8_t StopBits;
        uint_least8_t Reserved;
};
```

Vsa zaglavja imajo enako sestavljen prvi del (8 byte-ov), in sicer:

Start – začetek zaglavja (4 x FF).

HeaderTyp – tip zaglavja. V tem primeru 2, kar pomeni, da je to zaglavje, ki pove lastnosti povezave.

HeaderVer – verzija zaglavja. Pove, kaj je kateri podatek v zaglavju.

HeaderLen – dolžina zaglavja v ((#byte-ov / 4) – 1).

NextHeadLen – dolžina naslednjega zaglavja. Da program ve, kaj pričakovati, in če ne dobi zadostnega števila byte-ov, javi napako in ponovno resetira vezje. Če je enako nič, pomeni, da je to zadnje zaglavje.

Ostali podatki se razlikujejo med različnimi zaglavji. V tem primeru imamo naslednje:

BaudRate – hitrost povezave. Privzeta je 115200, lahko se z mostičkom določi 1000000.

DataBits – dolžina podatka v bitih. V tem primeru 8.

Parity – če imamo paritetni bit. V tem primeru ga ni.

StopBits – število stop bitov. V tem primeru eden.

Reserved – neuporabljeno. Da je število byte-ov deljivo s 4. Lahko se uporabi v bodoči verziji.

5.1.2 Inicializacija časovnikov TC

Časovniki so uporabljeni za dve funkciji, in sicer za štetje časa, preteklega med sprejemom dveh zaporednih byte-ov, ter za določanje, koliko časa bosta prižgani LED diodi, ki prikazujeta branje podatkov z linije med PC1 in PC2.

Za štetje časa med dvema zaporednima byte-oma so uporabljeni časovniki TC0, TC5 in TC1, za krmiljenje LED diod pa TC3 in TC4.

5.1.2.1 TC0, TC5 in TC1

```
void StartTimer0(void)
{
	cli();
	DDRD = (0<<DDD7);
	PORTD = (1<<PD7);
	TCNT0 = 0x00;
	OCR0A = 0x3F;
	TCCR0A = (1<<WGM01);
	TIMSK0 = (1<<OCIE0A);
	TCCR0B = (1<<CS01) | (1<<CS02);
	sei();
}
void StartTimer5(void)
{
	cli();
	DDRL = (0<<DDL2);
	PORTL = (1<<PE6);
	DDRL = (1<<PL3);
	TCNT5 = 0x0000;
	OCR5A = 0x007F;
	TCCR5A = (1<<COM5A1) | (1<<COM5A0) | (1<<WGM50);
	TCCR5B = (1<<WGM52) | (1<<CS51) | (1<<CS52);
	sei();
}
void StartTimer1(void)
{
	cli();
	DDRB = (1<<PB5);
	TCNT1 = 0x0000;
	OCR1A = 0x007F;
	TCCR1A = (1<<COM1A1) | (1<<COM1A0) | (1<<WGM10);
```

```
	TCCR1B = (1<<WGM12) | (1<<CS11);
	sei();
}
```

Ti trije časovniki so povezani med seboj tako, da tvorijo 24-bitni števec (uporabljenih 22 bitov). Časovnik TC1 šteje od 0 do FF. Pri tem tvori urin signal za časovnik TC5 na vmesniku PB5. Ko prešteje do FF, gre nazaj na 0. Pri tem se časovnik TC5 poveča za 1. Časovnik TC5 prav tako šteje od 0 do FF in prav tako tvori urin signal za časovnik TC0 na vmesniku PL3. Tako sem jih povezal zato, da ne porabljajo procesorskega časa za štetje. Ker štejejo samostojno, pomeni, da vedno preštejejo točno, kar je za projekt zelo pomembno. Časovnik TC0 šteje do 3F, torej prvih 6 bitov. Preostala dva bita sta uporabljena kot zastavice, ki računalniku PC3 povesta vrednost devetega bita trenutnega sprejetega byte-a ter izvor (računalnik PC1 ali PC2). Z dvaindvajsetimi biti, 16 MHz ter deljenjem urinega signala z 8 dobimo natančnost števcev 0,5 us, štejejo pa nekaj več kot dve sekundi (2,097152 s). Ko preštejejo do konca, se resetirajo ter začnejo šteti spet od 0 dalje. Ob resetu števcev na 0 se sproži prekinitev, v kateri se na računalnik PC3 pošlje 4 x »FF«. Tako računalnik PC3 vedno ve točno koliko časa je preteklo med dvema zaporednima bajtoma, tudi če je čas daljši od dveh sekund.

5.1.2.2 TC3 in TC4

```
void StartTimer3(void)
{
	cli();
	TCNT3 = 0x0000;
	OCR3A = 0x0100;
	TCCR3A = (1<<WGM32);
	TCCR3B = (1<<CS32);
	TIMSK3 = (1<<OCIE3A);
	sei();
}

void StartTimer4(void)
{
	cli();
	TCNT4 = 0x0000;
	OCR4A = 0x0100;
	TCCR4A = (1<<WGM42);
	TCCR4B = (1<<CS42);
	TIMSK4 = (1<<OCIE4A);
	sei();
}
```

Časovnika TC3, 4 sta uporabljena za krmiljenje LED diod, ki prikazujejo sprejemanje byte-ov. Oba delujeta enako, le da vsak krmili svojo LED diodo. Ob sprejetju byte-a se LED dioda prižge in se zažene časovnik TC3 ali TC4, odvisno, s katerega izvora je prispel byte. Ko časovnik prešteje do konca, sproži prekinitev, v kateri se LED dioda ugasne in števec ustavi. Števca sta

nastavljena tako, da štejeta 1,6 ms. Ta vrednost je bila izbrana po testiranju z različnimi vrednostmi, ker se je zdela najbolj primerna.

5.1.3 Inicializacija vmesnikov USART

5.1.3.1 USART0 (pošiljanje)

```
void InitUART_USB(void)
{
	UCSR0A = (0<<U2X0) | (0<<MPCM0);
	UCSR0B = (1<<RXCIE0) | (1<<RXEN0) | (1<<TXEN0);
	UCSR0C = (1<<UCSZ01) | (1<<UCSZ00);
	UBRR0 = 8;
}
```

Vmesnik USART0 je uporabljen za pošiljanje podatkov na računalnik PC3. Inicializirati ga je treba, preden se pošljejo zaglavja. Podatke pošilja preko čipa FT232RL. Na začetku se inicializira na hitrost 115200 bps in pošlje zaglavja. Za tem se lahko hitrost spremeni glede na postavitev stikal dip-switch.

5.1.3.2 USART1,2 (sprejemanje)

```
void InitUART_Receive(void)
{
	UCSR1A = (0<<U2X1) | (0<<MPCM1);
	UCSR2A = (0<<U2X2) | (0<<MPCM2);
	UCSR1B = (1<<RXCIE1) | (1<<RXEN1) | (1<<TXEN1) | (1<<UCSZ12);
	UCSR2B = (1<<RXCIE2) | (1<<RXEN2) | (1<<TXEN2) | (1<<UCSZ22);
	UCSR1C = (1<<UCSZ11) | (1<<UCSZ10);
	UCSR2C = (1<<UCSZ21) | (1<<UCSZ20);
	UBRR1 = 51;
	UBRR2 = 51;
}
```

Vmesnika USART1, 2 sta uporabljena za branje podatkov, ki se prenašajo med računalnikoma PC1 in PC2, zato morata biti nastavljena kot določa SAS protokol, ki se uporablja za prenos podatkov med računalnikoma PC1 in PC2. Poleg tega imajo omogočene prekinitve ob sprejetju byte-a (Rx Complete Interrupt Enable). To pa zato, da ob sprejetju vsakega byte-a sprožita prekinitev, v kateri se shranijo časovniki, sprejeti podatek in postavijo zastavice.

5.1.4 Prekinitvene funkcije

5.1.4.1 Prekinitvena funkcija TC0

```
ISR(TIMER0_COMPA_vect)
{
	cli();
	for(int i = 0;i < 4;i++)
	{
		while(!(CanSend()));
		UDR0 = 0xFF;
	}
	sei();

	if((UCSR1A >> 7) & 1)
	{
		cli();
		ReceivePC1();
	}
	else if((UCSR2A >> 7) & 1)
	{
		cli();
		ReceivePC2();
	}
}
```

Funkcija se sproži, ko časovnik TC0 prešteje do 3F, kar pomeni, da so vsi trije števci prešteli do konca. Funkcija pošlje na računalnik PC3 4 x FF, da ta točno ve, koliko časa je minilo od prejšnjega sprejetega byte-a. Na koncu še pregleda, če je medtem prispel novi byte na katerega od vhodov, ker so med pošiljanjem prekinitve onemogočene (cli()).

5.1.4.2 Prekinitveni funkciji vmesnikov USART1, 2

```
ISR(USART1_RX_vect)
{
	cli();
	ReceivePC1();
}

ISR(USART2_RX_vect)
{
	cli();
	ReceivePC2();
}
```

Ko prispe nov byte na vhodu vmesnika USART1/2, se sproži prekinitev, v kateri se kliče funkcija, ki shrani vse potrebne podatke v krožni medpomnilnik.

5.1.4.3 Prekinitvena funkcija časovnikov TC3, 4

```
ISR(TIMER3_COMPA_vect)
{
	cli();
	TCCR3B &= 0B11111000;
	PORTA = PORTA && 0b00000100;
	sei();
}

ISR(TIMER4_COMPA_vect)
{
	cli();
	TCCR4B &= 0B11111000;
	PORTA = PORTA && 0b00000010;
	sei();
}
```

Ob sprejetju novega byte-a se prižge LED dioda in zažene časovnik TC3/4. Ko časovnik TC3/4 prešteje do konca, se sproži prekinitev, v kateri se ugasne dioda in ustavi štetje časovnika. Moramo ga ustaviti, ker če ne, bi se vsake 1,6 ms sprožila prekinitev, tudi ko ni potrebe po tem, in s tem bi se porabljal procesorski čas.

5.1.5 Ostale funkcije

5.1.5.1 Main()

```
int main(void)//OK
{
	uint_least8_t DipSwitch = 0;
	uint_least32_t baudRate = 0;
	uint_least8_t tmpUBRR0 = 0;

	Tail = 0;
	Head = 0;
	prevLenght = 0;
	DDRA = (1 << DDA1) | (1 << DDA2);
	PORTA = 0;
	DDRF = 0b00000000;

	DipSwitch = PINF;

	switch (DipSwitch)
	{
	case 0:
		baudRate = 115200;
		tmpUBRR0 = 8;
		break;
	case 128:
```

```
		baudRate = 1000000;
		tmpUBRR0 = 0;
		break;
	}

	InitUART_USB();

	CopyHeader(&HdrSnifferVer);
	CopyHeader(&hdrFirmware);
	CopyHeader(&hdrHardware);

	HdrConnection.BaudRate = baudRate;
	CopyHeader(&HdrConnection);

	Send();
	while (!(UCSR0A & (1 << TXC0)));

	Head = 0;
	Tail = 0;

	//Inicializira Timerje
	StartTimer3();
	StartTimer4();
	StartTimer0();
	StartTimer5();
	StartTimer1();

	UBRR0 = tmpUBRR0;

	InitUART_RS232();

	while (1)
	{
		if (Head != Tail)
		{
			Send();
		}
	}
	return 0;
}
```

V glavni funkciji se najprej postavi indeksa Head in Tail na 0, da kažeta na prvi prostor v krožnem medpomnilniku, nastavi se pina PA1 in PA2 kot izhode za krmiljenje LED diod ter njihovo vrednost na 0, kar pomeni, da so diode ugasnjene. Port F se nastavi kot vhod za branje vrednosti stikal dip-switch. Po njihovi shranjeni vrednosti se nastavi vrednosti za določanje hitrosti povezave (UBRR0 in Baud Rate). Nato se najprej inicializira vmesnik USART0. Vmesnika USART1, 2 se inicializirata po poslanih zaglavjih zato, da ne bi med pošiljanjem zaglavij prišlo do kakšne prekinitve, če sta računalnika PC1, 2 že priključena. Za tem se prepiše vrednosti zaglavij v krožni medpomnilnik ter pošlje na računalnik PC3. Preden se v UBRR0 zapiše pravilna vrednost, funkcija v while zanki počaka, da se pošljejo vsi podatki na računalnik PC3. Če ne, bi se pri spremembi UBRR0 še ne poslani byte-i izgubili. Inicializirajo se časovniki

in nazadnje še vmesnika USART1, 2. Na koncu se začne izvajati neskončna while(1) zanka, v kateri se primerjata indeksa, in se kliče funkcija Send(), če sta različna.

5.1.5.2 CopyHeader()

```
void CopyHeader(uint_least8_t *Location)//OK
{
	uint_least8_t Lenght, i;

	i = 0;
	Lenght = ((*(Location + 6) + 1) * 4);

	while (i < Lenght)
	{
		Buffer[Head] = *Location;
		Location++;
		Head++;
		i++;
	}

	if (prevLenght > 0)
	{
		Buffer[Head - Lenght - prevLenght + 7] = (Lenght / 4) - 1;
	}

	prevLenght = Lenght;
}
```

Funkcija prepiše zaglavje v krožni medpomnilnik. Kot argument dobi naslov prvega byte-a v zaglavju. Nato se byte po byte-u premika naprej in zapisuje vrednost v medpomnilnik. Nazadnje shrani še dolžino zaglavja. Če se funkcija kliče ponovno za naslednje zaglavje, prebere dolžino (HeaderLen) in jo shrani v prejšnjega (NextHeadLen). Za to potrebuje dolžino prejšnjega, da lahko najde pravilno mesto v medpomnilniku, kamor shrani. Narejeno je tako, da se lahko v medpomnilnik shranijo zaglavja v poljubnem vrstnem redu ne glede na dolžino.

5.1.5.3 CanSend()

```
int CanSend(void)
{
	uint_least32_t counter = 1230769;

	while ((counter > 0) && (!( UCSR0A & (1<<UDRE0))))
	{
		counter--;
	}
	return ( UCSR0A & (1<<UDRE0));
}
```

Funkcija se kliče vsakič, ko hoče program poslati nov byte na računalnik PC3 preko vmesnika USART0. Funkcija približno eno sekundo pregleduje zastavico UDRE0 (Data Register Empty)

v registru UCSR0A, ki pove, če je vmesnik USART0 pripravljen na pošiljanje. Dokler je zastavica na 0, pomeni, da register še ni prazen. Po pretekli sekundi funkcija vrne vrednost zastavice = 0. Če se zastavica prej postavi na 1, funkcija skoči iz while() zanke in pošlje vrednost zastavice = 1, kar pomeni, da je vmesnik USART0 pripravljen na nov byte.

5.1.5.4 Send()

```
void Send(void)
{
	while(Tail != Head)
	{
		while(!(CanSend()));
		UDR0 = Buffer[Tail];
		Tail = ((Tail + 1) % BufferSize);
	}
}
```

Funkcija Send() pošilja podatke iz medpomnilnika na računalnik PC3. While(Tail != Head) zanka se izvaja, dokler indeksa Head in Tail nista enaka, kar pomeni, da ni več nobenih podatkov za pošiljanje. Nato pridemo v novo while(!(CanSend())) zanko, ki se izvaja, dokler funkcija CanSend() ne vrne vrednosti 1, kar pomeni, da je vmesnik USART0 pripravljen na nov byte. Pošlje se nov byte in poveča indeks Tail za 1 po modulu BufferSize, da kaže na naslednji byte za pošiljanje. Spreminja se po modulu, ker imamo krožni medpomnilnik in indeksa Head in Tail skočita iz zadnjega elementa v medpomnilniku na prvega.

5.1.5.5 ReceivePC1, 2

Ker sta funkciji RecivePC1() in ReceivePC2() enaki, je opisana samo ReceivePC1(). Razlikujeta se le v tem, da ena shranjuje podatke, ki prispejo na vmesnik USART1, druga pa podatke, ki prispejo na vmesnik USART2.

```
void ReceivePC1(void)
{
	Buffer[Head + 2] = TCNT1L;
	Buffer[Head + 1] = TCNT5L;
	Buffer[Head] = TCNT0;
	Head = Head + 3;

	PORTA = 0b00000010;
```

```
        TCNT3 = 0;
        TCCR3B = (1<<CS32) | (1<<CS30);

        TCNT1 = 0;
        TCNT5 = 0;
        TCNT0 = 0;

        if ((UCSR1B >> 1) & 1)
        {
                Buffer[Head - 3] = ((Buffer[Head - 3]) | (1L << (7)));
                Buffer[Head - 3] = ((Buffer[Head - 3]) & ~(1L << (6)));
        }
        else
        {
                Buffer[Head - 3] = ((Buffer[Head - 3]) & ~(1L << (7)));
                Buffer[Head - 3] = ((Buffer[Head - 3]) & ~(1L << (6)));
        }

        Buffer[Head] = UDR1;
        Head = ((Head + 1) % BufferSize);

        if((UCSR2A >> 7) & 1)
        {
                cli();
                ReceivePC2();
        }

}
```

Funkcija najprej shrani vrednosti časovnikov TC1, 5, 0. Vrstni red shranjevanja je pomemben. Shrani najprej vrednost časovnika TC1, ker šteje najhitreje, nato časovnika TC5 in nazadnje časovnika TC0, ki šteje najpočasneje. To pa zato, da preteče čim manj časa med prispelim podatkom in shranjevanjem časovnikov. S tem dobimo največjo natančnost. Po shranjevanju povečamo indeks Head na naslednji prazen prostor v medpomnilniku za shranjevanje naslednjega byte-a. Za tem se prižge LED dioda (vsaka funkcija prižge svojo diodo), postavi vrednost časovnika TC3/4 na 0 ter ga požene, da po določenem času (1,6 ms) ugasne diodo. Časovnike TC1, 5, 0 se postavi na 0, da začnejo šteti čas do naslednjega byte-a, postavi se zastavice ter shrani prispeli byte. Head indeks se spreminja enako kot Tail, tako da kroži po medpomnilniku. Na koncu se še preveri, če je v času izvajanja prispel nov byte na drugem vmesniku USART, in če je, se kliče pripadajoča funkcija.

Poglavje 6 SKLEPNE UGOTOVITVE

V projektu smo razvili vezje, ki prisluškuje komunikaciji dveh računalnikov in podatke skupaj s časom sprejetja pošlje na tretji računalnik, kjer se vse te informacije smiselno prikažejo. V ta namen smo razvili program, ki se izvaja na samem vezju, ter program, ki grafično prikazuje vse podatke skupaj z ostalimi informacijami, ki jim pripadajo.

Vezje je trenutno uporabno za prisluškovanje SAS on-line komunikaciji, za katero smo ga razvili. Uporablja se za razhroščevanje dela programa na igralnih avtomatih, ki skrbi za komunikacijo, ker smo pri uporabi igralnih avtomatov v igralnicah opazili, da ponekod prihaja do problemov, ki jih ni bilo mogoče rešiti z beleženjem. Vezje smo že testirali v realni situaciji in izkazalo se je kot uporabno. Ker do napak pri komunikaciji pride zelo redko, zaenkrat še nismo imeli sreče, da bi do tega prišlo medtem, ko smo imeli vezje priključeno.

Ker je uporabljena Arduino razvojna ploščica, se jo da hitro sprogramirati za prisluškovanje katerikoli drugi vrsti komunikacije. V podjetju že razmišljamo o novi verziji, ki bo prisluškovala protokolu CC-Talk. Ta protokol se uporablja za komunikacijo z določenimi bill-acceptorji, coin-acceptorji, hopperji in podobno, ki so sestavni del igralnih avtomatov.

Literatura

[1] (2014)8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. Dostopno na:

http://www.atmel.com/images/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[2] (2015) Future Technology Devices International Ltd. FT232R USB UART IC. Dostopno na:

http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[3] (2015) MAX220–MAX249 +5V-Powered, Multichannel RS-232 Drivers/Receivers. Dostopno na:

http://datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf

[4] (2016) Arduino MEGA 2560. Dostopno na:

https://www.arduino.cc/en/Main/ArduinoBoardMega2560